Multiple Indicator Multiple Cause (MIMC) Model with Discrete Indicators

Maria Kamran

1 Introduction

This document is intended for users of who want to implement multiple-indicator, multiple-cause (MIMC) modeling with discrete indicators in R. The potential target group includes advanced undergraduate students in technical fields, such as statistics or economics, graduate students in social sciences and engineering who are devising their own estimators, and researchers and practitioners. To use the function of described in this document, one should have basic knowledge of R language and is familiar enough with the concept of likelihood and maximum likelihood. If you are not familiar with the concepts of likelihood and maximum likelihood, please refer to the R vignette, "Maximum Likelihood Estimation with maxLik".

The organisation of the rest of this document is as follows: Section 2 summarises the preliminaries necessary to run the various functions outlined in this document. Section 3 gives a detailed account about estimation process including the econometric underpinnings, and further operations such as bootstrapping. Section 4 pertains to bug reports and feeback process, and 5 is the Licence. Details about the author are outlined in Section 6. Finally, Section 7 lists the references.

2 Installation

To run the functions the environment for data analysis is setup first, by configuring Java memory parameters and loading necessary libraries.

```
options(java.parameters="-Xmx5000m")
set.seed(300)
library(maxLik)
library(dplyr)
library(haven)
library(dummies)
library(xtable)
```

The `options(java.parameters="-Xmx5000m")` command increases the maximum heap size available to Java to 5000 MB, which is useful for handling large datasets or computationally intensive tasks. The `set.seed(300)` function

ensures reproducibility by setting a seed for random number generation. The script then loads several libraries: `systemfit` for initial values, `maxLik` for maximum likelihood estimation, `dplyr` for data manipulation, haven for importing and exporting data in various formats (like SAS, Stata, and SPSS), `dummies` for creating dummy variables, and `xtable` for converting data to LaTeX or HTML tables.

3 Usage

This section demonstrates how to utilize the code. It is divided into two components: the first provides a general intuition about the empirical procedure, followed by a step-by-step demonstration of its application to data. The practical application involves (i) loading and formatting the data, (ii) estimating initial values for maximum likelihood, (iii) running the MIMC function, and (iv) estimating residual correlations and conducting bootstrapping. The subsequent parts of this section follow this sequence.

3.1 Empirical Strategy

Following Jacoby & Mansuri's (2010) notation, let ζ represent the unobserved statehood. For the purpose of demonstration, it is assumed that there are two latent variables denoted by binary indicators y_o, where $o = 1$ and 2. A system of equations is estimated, with the oth equation being of form

$$y_o = 1(\beta_o \zeta + \mathbf{X}\xi_o + u_o > 0), \tag{1}$$

where $\beta_o > 0$, $\mathbf{X}$ is a matrix of exogenous characteristics. Here the error term u_o is i.i.d and is uncorrelated with the diagonal of a covariance matrix. The variable of interest is x_2, which has the following indicator function

$$\zeta = \eta x_2 + \epsilon. \tag{2}$$

Here ϵ is the error terms with variance σ_ϵ^2.[1] Substituting 2 into 1 leads to

$$y_o = 1(\eta_o x_2 + \mathbf{X}\xi_o + \nu_o > 0), \tag{3}$$

where y_o is a vector of the dependent variable, $\eta_o = \beta_o \eta$ and $\nu_o = \beta_o \epsilon + u_o$. Since the magnitude of η is not of direct interest, the null hypothesis ($\eta = 0$) is tested against $\eta < 0$, by looking at each of the η_o (since $\beta_o > 0$) for the two structural equations.

It is assumed that the variable of interest, x_2 a binary indicator suffers from endogeneity or selection problem, hence a 'first stage' equation is formalised

$$x_2 = 1(\mathbf{Z}\pi + \omega > 0), \tag{4}$$

[1] Without loss of generality $\mathbf{X}$ has been left out from equation 2.

where $\mathbf{Z}$ are the instruments and $\omega = \beta_w \epsilon + \Upsilon$ is the disturbance term.[2] The instruments and disturbance for the two reduced form equations o are uncorrelated.

For these estimations, ν_o and ω are assumed to be normally distributed, with equations 3 and 4 being simple probit conditional on ϵ. Tantamount to Jacoby & Mansuri (2010), ϵ is modelled nonparametrically using discrete factor approximation where coefficients are estimated jointly by maximum likelihood.

3.2 Data

To apply the empirical strategy of Section 3.1 to data, the follow step are taken:

```
1 WS <- read_dta("C:/Users/path")#Loading data
2 WS <- WS[complete.cases(WS),]# to remove observations with na
```

Listing 1: Loading Data

Line 1, uses the `read_dta` function from the `haven` package to load a dataset in `dta` format from the specified file path into a dataframe `WS`. Alternative methods are also compatible as long as the resulting data is in the form for a comprehensive data frame. Line 2, filters the dataset to retain only those rows that do not contain any missing values (NAs), ensuring the dataset is complete and ready for further analysis.

Once the data is loaded, the second step is to converts the all variables of interest into numeric vectors. This transformation is useful when the original variables are stored as factors, characters, or other non-numeric types.

```
1  y1 <- as.numeric(WS$y1)
2  y2 <- as.numeric(WS$y2)
3  x1 <- as.numeric(WS$x1)
4  x2 <- as.numeric(WS$x2)
5  x3 <- as.numeric(WS$x3)
6  z1 <- as.numeric(WS$z1)
7  z2 <- as.numeric(WS$z2)
8  d1 <- as.numeric(WS$d1)
9  d2 <- as.numeric(WS$d2)
10 d3 <- as.numeric(WS$d3)
```

Listing 2: Setting Data Vectors

Listing 2 shows that three possible types of variable. Based Section 3.1, `y1` and `y2` are the latent binary variables, `x2` is the endogenous variable with two instruments (`z1` and `z2`) and controls in form of `x1` and `x3` and the three fixed effects starting with the letter `d`. This can, of course, be adjusted to meet the specific needs of the user.

[2] The disturbance terms Υ and u_o are assumed to be mutually uncorrelated.

3.3 Maximum Likelihood

3.3.1 Intial Values

The first step for the estimation requires specifying the initial values. To do this Listing 3 sets up and estimates a system of simultaneous equations using the `systemfit` package.

```
eq1 <- y1 ~ x1 + x2 + x3 + d1 + d2 + d3
eq2 <- y2 ~ x1 + x2 + x3 + d1 + d2 + d3
eq3 <-  x2 ~ x1  + x3 + d1 + d2 + d3 + z1 + z2

system <- list(eq1=eq1,
               eq2=eq2,
               eq3=eq3)

aa <-
  systemfit(
    system,
    "3SLS",
    data = WS,
    inst =  ~ x1 + x3 + d1 + d2 + d3 + z1 + z2,
    pooled = TRUE,
    methodResidCov = "noDfCor",
    residCovWeighted = TRUE,
    method3sls = "GMM"
  )
```

Listing 3: Setting Intial Values Vectors

Here `system` is a list containing equations (`eq1`, `eq2`, `eq3`) to be estimated simultaneously. `aa` estimates this system using the `systemfit()` function with the following argumants:

- `system`: The list of equations (`eq1`, `eq2`, `eq3`). For the purpose of brevity, the system has three equations. Based on the requirement the list of equation can modified to include more equations.
- `3SLS`: Specifies the estimation method as Three-Stage Least Squares (3SLS). Refer to `systemfit` vignette to select other estimation methods.
- `inst`: Specifies the instrumental variables used `(x1, x3, d1, d2, d3, z1, z2)`.
- `pooled = TRUE`: Indicates pooled estimation.
- `methodResidCov = "noDfCor"`: Specifies the method for residual covariance estimation.
- `residCovWeighted = TRUE`: Indicates weighted residuals.
- `method3sls = "GMM"`: Specifies the method for 3SLS estimation as Generalized Method of Moments (GMM).

Once the coeffiencts are estimated using a linear estimation method, they are store in memory and renamed using the following command

```
names(aa$coefficients)[names(aa$coefficients) == "eq1_(Intercept)"]
    <- "b0"
```

The above snippet shows that the intercept of equation 1 denoted by `eq1_(Intercept)` is renamed as `b0`. For ease of interpretation meaningful variable names are assigned to the default ones.

3.3.2 Loglikelihood Function

Once the initial value for the parameters are estimated, a loglikelihood function `llr` is specified (see Section 8 for the full specification). For the purpose of simplicity, this section break the full function down step-by-step to understand its components and purpose.

1. **Parameter Assignment**: The function starts off by first doing a simple assignment operation that extracts the element from an indexed vector named `param` and assigns the name of the variable that will receive the value extracted. For instance intercept of `eq1` is the first element of vector `param`

```
    b0 <- param [1]
    ...
```

2. **Outer Sum Function**:

```
sum(
  log(
    ...  # Optimization functional form inside log function
  )
)
```

   - `sum(...)`: Computes the sum of all elements within the parentheses.
   - `log(...)`: Takes the natural logarithm of the expression inside.

3. **Optimization Functional Form**: The expression inside the log function is a weighted sum of three main components. Each component involves a combination of probabilities and indicator variables. Here's a high-level overview of the structure:

```
(exp(q1) / (exp(q1) + exp(q2) + 1)) * (...) +
(exp(q2) / (exp(q1) + exp(q2) + 1)) * (...) +
(1 - (exp(q1) / (exp(q1) + exp(q2) + 1)) - (exp(q2) / (exp(q1)
     + exp(q2) + 1))) * (...)
```

   Each of these components represents a probability weight multiplied by a product of multiple probabilities.

4. **Probability Weights**:

   - `(exp(q1) / (exp(q1) + exp(q2) + 1))`: The probability weight for *Component 1*.

- `(exp(q2) / (exp(q1) + exp(q2) + 1))`: The probability weight for the *Component 2*.
- `(1 - (exp(q1) / (exp(q1) + exp(q2) + 1)) - (exp(q2) / (exp(q1) + exp(q2) + 1)))`: The probability weight for the *Component 3*.

5. **Component 1**: This models for the case that the error terms (residuals) are independent of each other. In other words, residual covariances between equations is assumed to be zero. Hence the functional form in `(...)` is identical to that specified for the linear version (i.e., `eq1`,`eq2` and `eq3`).

```
(exp(q1) / (exp(q1) + exp(q2) + 1)) *
(
  ((pnorm(...))^y1) *
  ((pnorm(...))^y2) *
  ((pnorm(...))^x2) *
  ((1 - pnorm(...))^(1 - y1)) *
  ((1 - pnorm(...))^(1 - y2)) *
  ((1 - pnorm(...))^(1 - x2))
)
```

   Other features include

   - `pnorm(...)`: The cumulative distribution function of the normal distribution.

6. **Component 2**: The restriction on residual covariances is relaxed in this compnent and individual paramaters denoted by an indexed `rho` is included in each of the three equations.

```
    (exp(q2) / (exp(q1) + exp(q2) + 1)) *
(
  ((pnorm(... + rho1))^y1) *
  ((pnorm(... + rho2))^y2) *
  ((pnorm(... + rho3))^x2) *
  ((1 - pnorm(... + rho1))^(1 - y1)) *
  ((1 - pnorm(... + rho2))^(1 - y2)) *
  ((1 - pnorm(... + rho3))^(1 - x2))
)
```

7. **Component 3**: This case gives a residual factor-analytic covariance structure. That is residual covariance of each equation share a common variance denoted by `(exp(gamma) / (exp(gamma) + 1))`. Hence the rcode is as follows

```
(1 - (exp(q1) / (exp(q1) + exp(q2) + 1)) - (exp(q2) / (exp(q1)
     + exp(q2) + 1))) *
(
  ((pnorm(... + rho1 * (exp(gamma) / (exp(gamma) + 1))))^y1) *
  ((pnorm(... + rho2 * (exp(gamma) / (exp(gamma) + 1))))^y2) *
  ((pnorm(... + rho3 * (exp(gamma) / (exp(gamma) + 1))))^x2) *
  ((1 - pnorm(... + rho1 * (exp(gamma) / (exp(gamma) + 1))))
    ^(1 - y1)) *
```

```
  ((1 - pnorm(... + rho2 * (exp(gamma) / (exp(gamma) + 1))))
    ^(1 - y2)) *
  ((1 - pnorm(... + rho3 * (exp(gamma) / (exp(gamma) + 1))))
    ^(1 - x2))
)
```

3.3.3 Outer Function

To estimate the system, `maxLik_run` is designed to automate the process of maximizing the likelihood, `llr` using starting values and a specified optimization method.

```
   maxLik_run <- function(r, w, q, a, k, s, f) {
  tryCatch({

    llr <- function(param) {...}

    return(maxLik(
      llr,
      start = c(
        aa$coefficients,
        rho1 = r,
        rho2 = w,
        rho3 = q,
        rho4 = a,
        q1 = k,
        q2 = s,
        gamma = f
      ),
      method = "bfgs"
      ,
      control = list(
        tol = -1,
        reltol = 1e-12,
        gradtol = 1e-12
      )
    ))
  },
error = function(e)
  return(NA))
  }
```

Listing 4: Maximumlikelihood Function

Following is a breakdown of what each part of the code does:

1. **Function Definition**:
   - `maxLik_run` has seven arguments: `r`, `w`, `q`, `a`, `k`, `s`, and `f`. These arguments are the initial values for probability weights, residual covariances and varinance.
   - **Probability weights**: `k=q1` and `s=q2`.
   - **Residual Covariances**: Within the estimation `rho1`, `rho2` and `rho3` each represent the residual covariance in `eq1`, `eq2` and `eq3`, respectively. Within the function `maxLik_run`, `r=rho1`, `w=rho2` and `q=rho3`.

- **Residual variance**: In the estimation, the common residual variance is `(exp(gamma) / (exp(gamma) + 1))` where `gamma` is equals to the argument `f`.

2. **tryCatch Block:**:
   - The function is wrapped in a tryCatch block, which is used to handle potential errors that may occur during the execution of the function.
3. **maxLik Function**:

   Within `tryCatch`, the `maxLik` function is called:
   - llr : The likelihood function to be maximized.
   - `start`: Initial values for the parameters to start the optimization. `aa$coefficients` are coefficients from the linear model fitting, combined with the provided arguments (`r`, `w`, etc.).
   - `method`: Optimization method here is set at Broyden-Fletcher-Goldfarb-Shanno algorithm `"bfgs"`. Refer to the R vignette, "Maximum Likelihood Estimation with maxLik" for other options.
   - `control`: Control parameters for the optimization process. Here `tol`, `reltol`, `gradtol` specify convergence criteria.

3.3.4 Final Execution

1. **Defining the Grid**: Selection of the definitive arguments for `maxLik_run` is an iterative process. To automate this, first `expand.grid` is used to create a data frame to generate all possible combinations of the arguments.

```
  st <-
expand.grid(
  rho1 = c(-0.09,1),
  rho2 = c(1.251976),
  rho3 = c(0.5),
  q1 = c(0.1),
  q2 = c(-3.42),
  gamma = c(-3.26)
)
```

Listing 5: Grid Search

2. **Applying the Function**: The `maxLik_run` Function is executed via the command `Map`. Depending on the dimension of the `st`, executing this function may be a time consuming process.

```
maxLik_list <-
  Map(maxLik_run,
      st$rho1,
      st$rho2,
      st$rho3,
      st$q1,
      st$q2,
```

```
8       st$gamma)
```

Listing 6: Final Implementation

The resulting `maxLik_list` is a list of lists where section of final specification needs to be made manually based on standard statistical criterion.

3.4 Other Operations

Besides obtaining point estimates, the following operation can also be performed once the final estimation is selected:

- Exporting the estimates to latex via `latex` function.
- Estimating residual covariances using `rcor` and their standard errors
- Bootstrapping the estimates.

4 Bug Reports and Contributions

I welcome contributions and feedback from the community to help improve the functions. If you encounter any bugs or issues, please report them through our GitHub Issue Tracker. When reporting a bug, provide as much detail as possible, including the steps to reproduce the issue and any relevant error messages. If you would like to contribute to the development of the package, you can fork my repository on GitHub at `https://github.com/mariakamran86/SystemMaxLik`, make your changes, and submit a pull request. I appreciate your efforts to enhance the functions and will review all contributions promptly.

5 License

AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

6 Author

Maria Kamran is an experienced data scientist and economist, specializing in the application of R for data analysis and statistical modeling. She is in her final year of Ph.D. in Economics at the Institut de hautes études internationales et du développement and has over five years of professional experience in academia.

7 References

Jacoby, H. G., & Mansuri, G. (2010). Watta Satta: Bride exchange and women's welfare in rural Pakistan. American Economic Review, 100(4), 1804-1825.

8 Appendix

```
llr <- function(param) {

#Parameters----
b0  <-  param[1]
b1  <-  param[2]
b2  <-  param[3]
b3  <-  param[4]
bl1  <-  param[5]
bl2  <-  param[6]
bl3  <-  param[7]

c0  <-  param[8]
c1  <-  param[9]
c2  <-  param[10]
c3  <-  param[11]
cl1 <-  param[12]
cl2 <-  param[13]
cl3 <-  param[14]

g0 <- param[15]
g1 <- param[16]
g3 <- param[17]
gz1 <- param[18]
gz2 <- param[19]
gl1  <-  param[20]
gl2  <-  param[21]
gl3  <-  param[22]

q1 <- param[23]
q2 <- param[24]
rho1 <- param[25]
rho2 <- param[26]
rho3 <- param[27]
gamma <- param[28]


```

```
  #ML Function----
sum(log(
    (exp(q1) / (exp(q1) + exp(q2) + 1)) *
    ((pnorm(b0 + b1 * x1 + b2 * x2 + b3 * x3 + bl1 * d1 + bl2 * d2 + bl3 * d3))^y1 *
      (pnorm(c0 + c1 * x1 + c2 * x2 + c3 * x3 + cl1 * d1 + cl2 * d2 + cl3 * d3))^y2 *
      (pnorm(g0 + g1 * x1 + g3 * x3 + gl1 * d1 + gl2 * d2 + gl3 * d3 + gz1 * z1 + gz2 * z2))^x2 *
      (1 - pnorm(b0 + b1 * x1 + b2 * x2 + b3 * x3 + bl1 * d1 + bl2 * d2 + bl3 * d3))^(1 - y1) *
      (1 - pnorm(c0 + c1 * x1 + c2 * x2 + c3 * x3 + cl1 * d1 + cl2 * d2 + cl3 * d3))^(1 - y2) *
      (1 - pnorm(g0 + g1 * x1 + g3 * x3 + gl1 * d1 + gl2 * d2 + gl3 * d3 + gz1 * z1 + gz2 * z2))^(1 - x2))
    +
    (exp(q2) / (exp(q1) + exp(q2) + 1)) *
    ((pnorm(b0 + b1 * x1 + b2 * x2 + b3 * x3 + bl1 * d1 + bl2 * d2 + bl3 * d3 + rho1))^y1 *
      (pnorm(c0 + c1 * x1 + c2 * x2 + c3 * x3 + cl1 * d1 + cl2 * d2 + cl3 * d3 + rho2))^y2 *
      (pnorm(g0 + g1 * x1 + g3 * x3 + gl1 * d1 + gl2 * d2 + gl3 * d3 + gz1 * z1 + gz2 * z2 + rho3))^x2 *
      (1 - pnorm(b0 + b1 * x1 + b2 * x2 + b3 * x3 + bl1 * d1 + bl2 * d2 + bl3 * d3 + rho1))^(1 - y1) *
      (1 - pnorm(c0 + c1 * x1 + c2 * x2 + c3 * x3 + cl1 * d1 + cl2 * d2 + cl3 * d3 + rho2))^(1 - y2) *
      (1 - pnorm(g0 + g1 * x1 + g3 * x3 + gl1 * d1 + gl2 * d2 + gl3 * d3 + gz1 * z1 + gz2 * z2 + rho3))^(1
    - x2)) +
    (1 - (exp(q1) / (exp(q1) + exp(q2) + 1)) - (exp(q2) / (exp(q1) + exp(q2) + 1))) *
    (
      (pnorm(b0 + b1 * x1 + b2 * x2 + b3 * x3 + bl1 * d1 + bl2 * d2 + bl3 * d3 + rho1 * (exp(gamma) / (exp(
    gamma) + 1))))^y1 *
      (pnorm(c0 + c1 * x1 + c2 * x2 + c3 * x3 + cl1 * d1 + cl2 * d2 + cl3 * d3 + rho2 * (exp(gamma) / (exp(
    gamma) + 1))))^y2 *
      (pnorm(g0 + g1 * x1 + g3 * x3 + gl1 * d1 + gl2 * d2 + gl3 * d3 + gz1 * z1 + gz2 * z2 * rh3 * (exp(
    gamma) / (exp(gamma) + 1))))^x2 *
      (1 - pnorm(b0 + b1 * x1 + b2 * x2 + b3 * x3 + bl1 * d1 + bl2 * d2 + bl3 * d3 + rho1 * (exp(gamma) / (
    exp(gamma) + 1))))^(1 - y1) *
      (1 - pnorm(c0 + c1 * x1 + c2 * x2 + c3 * x3 + cl1 * d1 + cl2 * d2 + cl3 * d3 + rho2 * (exp(gamma) / (
    exp(gamma) + 1))))^(1 - y2) *
      (1 - pnorm(g0 + g1 * x1 + g3 * x3 + gl1 * d1 + gl2 * d2 + gl3 * d3 + gz1 * z1 + gz2 * z2 + rho3 * (
    exp(gamma) / (exp(gamma) + 1))))^(1 - x2)
    ))}
```

Listing 7: Maximum Likelihood Function